LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

VISUAALISEN SISÄLLÖN LUONTI OGRE-PELIMOOTTORILLE

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Mediatekniikan koulutusohjelma
Teknisen visualisoinnin suuntautumis-
vaihtoehto
Opinnäytetyö
Kevät 2012
Hannu Santala

Lahden ammattikorkeakoulu
Mediatekniikan koulutusohjelma

SANTALA, HANNU: Visuaalisen sisällön luonti OGRE-pelimoottorille

Teknisen visualisoinnin opinnäytetyö, 38 sivua

Kevät 2012

TIIVISTELMÄ

Opinnäytetyössä käydään läpi OGRE-pelimoottorin ominaisuuksia ja perustietoja kyseisestä moottorista. Myös kehittäjiin vaikuttavia muita tekijöitä tarkastellaan, kuten lisenssejä ja niiden vaikutusta kehittämiseen. Teoriaosuudessa tarkastellaan myös kolmea muuta pelimoottoria, Unreal Enginea, Unitya ja CryEnginea, ja niiden eroavaisuuksia OGRE:en verrattuna. Työssä käydään läpi myös OGRE:n ominaisuuksia hieman tarkemmin ja sitä, millaisia ohjelmia tarvitaan, jotta OGRE:n avulla voidaan luoda visuaalisesti näyttäviä sovelluksia. Myös OGRE:en liittyviä työkaluja, kuten suosituimpia editoreja ja liitännäisiä tarkastellaan opinnäytetyössä.

Opinnäytetyö sisältää myös case-osan, jossa luodaan sovellus, jonka on tarkoitus toimia esittely-ympäristönä 3D-malleille ja muille töille. Sovelluksesta yritettiin luoda mahdollisimman visuaalisesti näyttävä, ilman että se käyttäisi liikaa tietokoneen resursseja. Sovelluksen on myös tarkoitus selvittää, toimiiko OGRE otsikon mukaan pelimoottorina vai onko se rajoitettu lähinnä graafisiin sovelluksiin. Case-projektin sovelluksen luonnissa pyritään käyttämään pääasiassa OGRE:lle saatavissa olevia työkaluja.

Asiasanat: OGRE, Ogre3d, pelimoottori, grafiikkamoottori, visualisointi

Lahti University of Applied Sciences
Degree Programme in Media Technology

SANTALA, HANNU: Creation of Visual Content for OGRE Game Engine

Bachelor's Thesis in Visualization Engineering, 38 pages

Spring 2012

ABSTRACT

This Bachelor's thesis deals with the features and basic information about the OGRE game engine. It also inspects other factors affecting developers, like licenses and their impact on developing. Three other game engines, Unreal Engine, Unity, and CryEngine, are examined in the theory part of the thesis and compared to OGRE. In addition to comparing the engines to OGRE, special attention is paid to the features of OGRE and to what makes this engine perform at its best to create visually impressive applications. The tools related to OGRE, like the most popular editors and plug-ins, are also examined in the thesis.

The thesis also contains a case part where an application is created which is meant to work as a presentation environment for 3D models and other works. This application was created as visually impressive as possible without using too many computer resources. The purpose of the application is also to determine whether OGRE can be used as a game engine, as the title suggests, or only for graphical applications. The application in the case project was created mostly by using tools available for OGRE.

Key words: OGRE, Ogre3d, game engine, graphics engine, visualization

SISÄLLYS

1 JOHDANTO

Pelialalla on monenlaisia pelimoottoreita ja niitä käytetään moniin eri tarkoituksiin. Jotkin niistä on tarkoitettu vain yhden alustan pelisuunnittelua varten, kun osalla voi luoda pelejä monille eri alustoille. Monet kaupallisista peleistä käyttävät hyödykseen jo entuudestaan hyväksi todettuja ja tunnettuja pelimoottoreita, sillä tuntemattomampien moottorien käyttö voi olla riski pelinkehittäjille. Monilla kehittäjillä ei myöskään ole aikaa eikä varaa valmistaa heille omaa pelimoottoria, joten heidän pitää valita valmiiksi kehitetty moottori pelilleen. Usein näissä tapauksissa valitaan pelille juuri jokin tunnettu moottori, sillä niillä on yleensä suuri yhteisö, jonka avulla on saatu paljon tukea ja dokumentaatiota kehittäjille.

On kuitenkin olemassa myös pelimoottoreita, jotka eivät ole saaneet kovin suurta käyttäjämäärää, vaikka moottori sisältäisi melkein samat ominaisuudet kuin tunnetut moottorit. Usein nämä pelimoottorit ovat myös käyttäjilleen ilmaisia toisin kuin moottorit, joita käytetään paljon.

OGRE on monella tavalla erilainen kuin niin sanotut tavalliset pelimoottorit. Sen perusominaisuuksiin ei kuulu kuin grafiikan esittäminen, mikä tekee siitä poikkeavan muihin moottoreihin nähden.

Opinnäytetyön teoriaosuudessa tutkitaan OGRE-pelimoottoria lähemmin. Siinä yritetään selvittää eroavaisuuksia muutamaan muuhun pelimoottoriin nähden. Teoriaosiossa käydään läpi myös, minkälaisia liitännäisiä ja muita ominaisuuksia OGRE:een on mahdollista lisätä pelkkien perusominaisuuksien ympärille. Case-osuudessa koetetaan käydä läpi, minkälaista visuaalista sisältöä OGRE:lla ja siihen liittyvillä työkaluilla on mahdollista luoda.

2 OGRE

2.1 Tausta

OGRE on 3D-grafiikkamoottori, joka on tehty käyttäen C++-ohjelmointikieltä. Se on yritetty tehdä intuitiiviseksi, jotta kehittäjien olisi mahdollisimman helppo tehdä graafisia sovelluksia. OGRE on lyhenne sanoista Object-Oriented Graphics Rendering Engine, mikä viittaa sen oliopohjaisuuteen. (Ogre3D 2011c.)

OGRE:n kehittäminen alkoi vuonna 2000, kun Steve Streeting halusi luoda hyvin suunnitellun ja itsenäisen moottorin. Tätä ennen hänellä oli ollut DIMClass-projekti, jonka tarkoitus oli sama kuin OGRE:lla, mutta hän lopetti sen kehittämisen, sillä DIMClass oli tarkoitettu vain Direct3D-pohjaiseksi. (Ogre3D 2010a.)

Vuonna 2005 julkaistiin OGRE:n versio 1.0.0, joka oli OGRE:n ensimmäinen virallinen versio. Se tunnetaan myös koodinimellä Azathoth, ja sen ominaisuuksiin kuului muun muassa HDRI-tuki, CEgui sekä mallien vienti XSI-ohjelmasta OGRE:een. (Ogre3D 2010a.)

Viimeisin Cthugha-niminen versio on versionumeroinniltaan 1.7, joka julkaistiin helmikuussa 2010. Eräs suurimmista kyseiseen versioon muuttuneista asioista oli lisenssin vaihtuminen GNU Lesser General Public ja OGRE Unrestricted -lisensseistä MIT-lisenssiin. (Ogre3D 2012a.)

2.2 Käyttötarkoitukset

OGRE:n ainoa tarkoitus on saada aikaiseksi reaaliaikaista grafiikkaa tietokoneen ruudulle. Se ei pysty tuottamaan minkäänlaisia ääniä, hallitsemaan verkkoyhteyksiä tai ottamaan vastaan käyttäjän komentoja. Kuten OGRE:n toinen nimi Ogre3D kertoo, sillä ei tehdä muuta kuin esitetään 3D-grafiikkaa. (Junker 2006, 3.)

Kuitenkin OGRE voidaan yhdistää muihin komponentteihin, jolloin sovelluksiin voidaan saada esimerkiksi ääntä. Kun OGRE:a käytetään tällä tavalla, voidaan sitä käyttämällä luoda esimerkiksi pelejä tai interaktiivisia simulaatioita. (Junker 2006, 3.)

Jos sovelluksessa halutaan kuitenkin käyttää pelkkää OGRE:a, ovat käyttökohteet melko rajatut. Koska OGRE:lla voi luoda vain grafiikkaa, kyseisissä sovelluksissa ei voisi olla minkäänlaista vuorovaikutusta, eikä myöskään ääniä. Kuitenkin mahdollisia käyttökohteita voisivat olla esimerkiksi erilaiset esittely- ja demovideot, joita voisi käyttää tuotteiden esittelyyn messuilla ja muissa vastaavissa tilaisuuksissa. Hyvä puoli normaaleihin videoihin verrattuna olisi kohtuullisen helppo muokattavuus ja reaaliaikaisuus. Esimerkiksi jos muuttaisi kameran reittiä, täytyisi normaalisti koko video renderöidä uudestaan, missä kestäisi erityisesti korkeilla kuvanlaatuasetuksilla erittäin kauan. OGRE:n sovelluksessa riittäisi vain sovelluksen uudelleen kääntäminen ja tämän jälkeen sen voisi ajaa reaaliajassa.

Kuitenkin normaalissa videossa on mahdollista saada parempaa kuvanlaatua, jopa huonoimmilla kokoonpanoilla. Jos OGRE:lla yrittää ajaa kohtausta, missä on paljon yksityiskohtia, ei sitä välttämättä pysty ajamaan tasaisella ruudunpäivitysnopeudella, vaikka käytössä olisi kuinka tehokas tietokone. Normaaleissa valmiiksi renderöidyissä videoissa tämä ei tietenkään ole ongelma.

2.3 Lisenssit

OGRE:n ensimmäinen versio perustui GNU Lesser General Public License -lisenssiin. GNU Lesser General Public License eli LGPL on kompromissi niin kutsuttujen vahvojen copyleft-lisenssien ja enemmän sallivien lisenssien välillä. Se on hieman muokattu versio GNU General Public Licensesta, ja merkittävin ero näiden välillä on se, että LGPL:n saa liittää muihinkin kuin pelkästään GPL- ja LGPL-lisensseihin. (Free Software Foundation 2011; Ogre3D 2012a.)

OGRE:n versiossa 1.4.0 otettiin LGPL:n rinnalle myös OGRE:n oma OUL, OGRE ”Unrestricted” License. OUL otettiin käyttöön lähinnä sen takia, että sovellusten kehitys ja julkaiseminen pelikonsoleille olisi helpompaa. OUL:ää käyttämällä saatiin täytettyä julkaisijoiden vaatimuksia kuten esimerkiksi salassapitosopimukset. (Ogre3D 2012a.)

LGPL ja OUL kuitenkin korvattiin MIT-lisenssillä OGRE:n 1.7.0-versiossa. MIT-lisenssi on nimetty kehityspaikkansa Massachusettsin teknillisen korkeakoulun mukaan. Se sallii ohjelmistojen ja koodin uudelleenkäytön myös maksullisissa

sovelluksissa, kunhan lisenssin teksti, jossa ehdot määritellään, säilyy lähdekoodissa. Kun sovelluksen julkaisee kyseisen lisenssin alaisena, ei lähdekoodia tarvitse julkaista. Koodia saa myös muokata, kopioida tai käyttää toisessa projektissa. MIT-lisenssi on myös yhteensopiva GPL:n kanssa, minkä takia GPL- ja MIT-lisensoituja sovelluksia voidaan yhdistää. (Ogre3D 2012a, The Linux Information Project 2009.)

Alla olevasta taulukosta voi nähdä, millä tavoin eri lisenssit vaikuttavat koodin ja sovelluksen jakeluun ja kehittämiseen.

TAULUKKO 1. Lisenssien vaikutus koodiin ja sovelluksen jakeluun ja kehittämiseen (Rusin 2009.)

Lisenssi	**Maksullisiin sovelluksiin yhdistäminen**	**Sovelluksen jakelu**	**Koodin uudelleenjakaminen muutosten jälkeen**
GPL	Ei sallittu	Sallittu vain GPL-yhteensopivien lisenssien kanssa	Vain jos johdannaisen lisenssi on GPL
LGPL	Sallittu	Sallittu, kun LGPL.materiaalin lähdekoodi on julkaistu	Vain jos johdannaisen lisenssi on GPL tai LGPL
MIT	Sallittu	Sallittu	Sallittu

3 PELIMOOTTORIEN VERTAILU

3.1 OGRE verrattuna muihin pelimoottoreihin

Vaikka OGRE ei olekaan suoranaisesti pelimoottori, silti monet tahot käyttävät sitä pelien graafisen puolen tekemiseen. OGRE:n kannalta se on sekä hyvä että huono asia, että sillä voi tehdä vain pelien grafiikat.

Jos OGRE:ssa olisi mahdollisuus luoda muutakin, voisi se olla paljon vaikeakäyttöisempi kuin mitä se nyt on. Myös SDK:n koko on täten pysynyt siedettävämmän kokoisena, kun sinne ei ole tarvinnut lisätä muuta kuin graafisiin asioihin liittyvät komponentit. Näin ollen kehittäjät saavat itse valita grafiikoiden ympärille haluansa asiat, eikä tarvitse tyytyä tiettyihin ennalta määrättyihin komponentteihin. OGRE on myös täysin ilmainen, mikä johtuu siitä, ettei siihen ole lisätty ominaisuuksia, jotka tarvitsisivat ulkopuolisia, maksullisia komponentteja.

Monet pelikehittäjät kuitenkin haluavat mahdollisimman kattavan kokonaisuuden, johon ei tarvitse itse etsiä jokaista erillistä liitännäistä. Tämän takia monet muut pelimoottorit ovat erittäin suosittuja harrastajien ja myös ammattilaisten keskuudessa.

3.2 Unreal Engine

Unreal Engine on Epic Gamesin vuonna 1998 julkaistu pelimoottori. Sen ominaisuudet alkuaikoina rajoittuivat renderöintiin, törmäyksen tarkastukseen, tekoälyyn ja muutamaan muuhun perusominaisuuteen. Unreal Enginen alku oli myös hankala, sillä Epic Games panosti suuresti 3dfx:n Glide-teknologiaan, mikä osoittautui virheeksi, sillä 3dfx ajautui konkurssiin ja Glide poistui markkinoilta. Moottoriin saatiin kuitenkin paranneltua OpenGL- ja Direct3D-tukea, ja siinä oli mukana UnrealScript-skriptauskieli, jonka avulla Unreal Engine sai kohtuullisen suuren mod-käyttäjäkunnan. (Wikimedia Foundation 2012b.)

Usean vuoden kehityksen tuloksena Unreal Enginen kolmas versio on nyt yksi markkinoiden käytetyimmistä pelimoottoreista. Unreal Engine 3:sen ohella Epic Games julkaisi myös Unreal Development Kitin, joka tarjoaa erityisesti harrasta-

jille ja pienemmille yrityksille mahdollisuuden käyttää Unreal Enginea huokeampaan hintaan. Sitä voi käyttää täysin ilmaiseksi, jos ohjelmasta ei aiota tehdä kaupallista, ja jos maksaa 99 dollarin lisenssimaksun saa UDK:n avulla tehdä myös kaupallisia ohjelmia. Jos ohjelmaa saadaan myytyä niin paljon, että sen avulla saadaan yli 50 000 dollarin voitot, joutuu Epic Gamesille maksamaan kaikesta 50 000 yli menevästä summasta 25 %. (Epic Games 2010.)

Unreal Enginen uusin versio on saatavilla suurimmalle osalle suosituista alustoista, kuten Windows, Xbox 360, PlayStation 3, Mac OS X, iOS, Linux, Android ja Adobe Flash Player 11. Sen grafiikkamoottorin tärkeimpiin ominaisuuksiin kuuluu dynaamiset varjot, 64-bittinen HDR-valaistus ja tuki Direct3D 11:n ominaisuuksille. (Epic Games 2010; Epic Games 2012.)

KUVA 1. Vertailu Unreal Enginen versioiden 1 - 3 välillä (Wikimedia Foundation 2006.)

3.3 Unity

Unity Technologies julkaisi Unity-pelimoottorin ensimmäisen kaupallisen version vuonna 2005. Unitya pidetään enimmäkseen pelintekoon kohdistuvana työvälineenä, mutta sillä tehdään myös muitakin ohjelmia, kuten interaktiivisia arkkitehtuurivisualisointeja tai 3D-animaatioita. Se on monien indie-pelinkehittäjien suo-

siossa kohtalaisen halvan hinnan ja suuren ominaisuusmäärän takia. (Unity Technologies 2005.)

Unitysta on tarjolla kaksi eri versiota: ilmainen Unityn perusversio ja 1500 dollaria maksava Unity Pro. Suurin ero näiden kahden version välillä on ominaisuuksien määrä. Vaikka Unityn perusversio riittääkin monelle harrastelijalle ja sillä pystyy tekemään kaikki perustoiminnot, on siitä silti jätetty monia ominaisuuksia pois, jotka löytyvät Pro-versiosta. Näihin ominaisuuksiin kuuluu muun muassa videoiden toisto ja streamaus eli suoratoisto, global illumination eli epäsuoravalaistus, render-to-texture-efektit ja korkealaatuinen reaaliaikainen varjostus. (Unity Technologies 2011.)

Koska Unitylla on mahdollista tehdä ohjelmia monelle eri alustalle, on sille olemassa myös maksullisia lisäosia. iOS- ja Android-mobiilialustoille on 400 dollaria maksavat lisäosat, mutta saatavilla on myös Pro-lisäosat molemmille alustoille, jotka sisältävä enemmän ominaisuuksia, toimivat vain Unity Pron kanssa ja maksavat 1500 dollaria. Mobiilikehittäjien kannalta tämä on huono asia, sillä Unitysta ei ole mobiilialustoille ilmaisversiota saatavilla. Kaikkiaan Unitylla pystyy tekemään ohjelmia Windowsille, Mac OS X:lle, Xbox 360:lle, PlayStation 3:lle, Wii:lle, iOS:lle, Androidille ja selaimille, jotka tukevat Unityn liitännäistä. (Unity Technologies 2011; DevMaster 2011b.)

3.4 CryEngine

CryEnginen ensimmäinen versio kehitettiin vuonna 2004 teknologiademoksi näytönohjainvalmistaja Nvidialle. Demon valmistanut Crytek huomasi siinä kuitenkin potentiaalia myös muuhun, ja he kehittivät sen avulla FarCry-nimisen pelin. FarCry ja Crytekin muut teokset ovat olleen tarpeeksi suosittuja, että he ovat päässeet CryEnginen kolmanteen versioon. (Wikimedia Foundation 2012a.)

CryEngine 3 saatiin valmiiksi vuonna 2009 Crytekin uutta Crysis 2 -peliä varten. Muutaman kuukauden kuluttua moottorin valmistumisesta julkaistiin myös ilmaisversio CryEnginen kolmannesta versiosta, joka oli tarkoitettu ei-kaupallisiin sovelluksiin. Niin sanottu ilmainen versio on saatavilla myös itsenäisille kehittäjille, joilla ei ole julkaisijaa pelilleen. He eivät joudu maksamaan mitään itse peli-

moottorista, vaan joutuvat jakamaan osan voitoista Crytekin kanssa. Maksettava osuus on 20 % kaikista CryEnginen avulla tehtyjen ohjelmien tuotoista. (Crytek 2012.)

CryEngine 3 on saatavilla Windows-, Xbox 360-, PlayStation 3-alustoille. Se tulee myös tukemaan Wii U -alustaa kyseisen konsolin julkaisun jälkeen. Rajapinnoista CryEngine tukee tällä hetkellä Windows-käyttöjärjestelmän puolella DirectX 9:ää, DirectX 10:tä ja DirectX 11:tä. Myös Xbox 360 tukee DirectX 9:ää. Muut alustat eivät voi käyttää DirectX:ää, sillä se on Microsoftin omaisuutta. Tästä syystä PlayStation 3 ja Wii U käyttävät OpenGL ES:ää. (Develop 2011; DevMaster 2011a.)

TAULUKKO 2. Vertailutaulukko neljän eri pelimoottorin välillä (DevMaster 2011a; DevMaster 2011b; DevMaster 2011c; DevMaster 2011d.)

	OGRE	Unreal Engine	Unity	CryEngine
Windows	+	+	+	+
Mac OS X	+	+	+	-
Linux	+	-	+	-
Xbox 360	+	+	+	+
PlayStation 3	+	+	+	+
Wii	+	+	+	-
iOS	+	+	+	-
Android	-	+	+	-
DirectX 10	+	+	+	+
DirectX 11	-	+	-	+
OpenGL	+	+	+	+
Ohjelmointikieli	C, C++	C, C++	JavaScript, C#	C, C++
Hinta harrastelijakäytössä	Ilmainen	Ilmainen	Ilmainen	Ilmainen
Hinta kaupallisessa käytössä	Ilmainen	Maksullinen	Ilmainen rajoituksin	Rojaltimaksullinen

4 OGRE:N OMINAISUUDET

4.1 Meshit ja geometria

Meshit ovat näkyviä tai joskus myös näkymättömiä objekteja 3D-sovelluksessa. Ne on luotu yleensä esittämään asioita, joita sovelluksen tekijät pitävät tärkeinä elementteinä sovelluksessa tai sen yleisilmeen kannalta. Meshit koostuvat yleensä kolmioista, ja sillä voi olla myös jokin tekstuuri liitettynä siihen. Kyseisiä elementtejä voivat olla esimerkiksi maasto, rakennukset, hahmot tai puut. (Ogre3D 2011d.)

OGRE käyttää meshien käsittelyyn omaa XML-tiedostorakennetta. Se on ainoa tapa, jolla 3D-malleja voidaan tuoda OGRE:a käyttävään sovellukseen. Tästä syystä mallintamisohjelmistoille on luotu exporter-plugineita, joiden avulla tietyillä ohjelmistoilla tehdyt mallit pystytään muuttamaan OGRE:n käyttämään tiedostomuotoon. (Ogre3D 2011b.)

OGRE käyttää malleille kahta eri tiedostomuotoa, jotka on tarkoitettu hieman eri tarkoituksiin. Tiedostomuoto, jota OGRE-sovellukset käyttävät, on .mesh-muoto. Se on binäärimuoto, joka soveltuu vain sovelluksien sisäiseen käyttöön, eikä kyseistä muotoa pysty muokkaamaan, ellei sitä ensin muuta muuhun muotoon. .mesh-muodon pystyy muuttamaan .mesh.xml-muotoon muunnostyökaluilla kuten OgreXmlConverterilla. .mesh.xml on siis muokattavissa oleva muoto, jossa kaikki kappaleeseen liittyvä tieto on luettavissa xml-formaatissa. Tietyt exporter-pluginit osaavat myös luoda .mesh.xml-muotoa suoraan mallintamisohjelmistoista .mesh-muodon lisäksi. .mesh.xml-tiedostoja ei pysty käyttämään OGRE-sovelluksissa, vaan se pitää ensin muuttaa .mesh-muotoon. (Ogre3D 2011d.)

OGRE:n meshit ovat niin kutsuttuja progressiivisia meshejä, joten niillä on Level Of Detail -ominaisuus. Tämän avulla pystytään säätämään meshin yksityiskohtien tarkkuutta, ilman että joudutaan luomaan suurta määrää erilaisia malleja. Esimerkiksi jos tiettyjä malleja katsoo tarpeeksi kaukaa OGRE-sovelluksessa, muuttaa grafiikkamoottori mallin vähemmän tietokonetta rasittavaan muotoon, ilman että tätä muutosta pystyisi huomaamaan kovin helposti suuren etäisyyden takia. (Ogre3D 2011b.)

Meshit tukevat myös hahmoanimaatioita, joten hahmojen animointiin voi käyttää OGRE:n käyttämiä hahmoanimaatiomalleja. Hahmoanimaatiot saavat myös sulautumaan useamman eri animaation kesken, minkä avulla animaatiosta toiseen siirtyminen on helpompaa ja luonnollisemman näköistä. OGRE mahdollistaa myös luiden painotuksen muuttamisen hahmoanimaatioissa. (Ogre3D 2011b.)

OGRE:n meshit ovat kaikkiaan monipuolisia, joustavia ja sisältävät paljon erilaisia muokattavia muuttujia. Meshit voivat saada monia erilaisia tietomuotoja, mutta pääasiassa ne koostuvat kahdesta eri osasta: vertex bufferista ja index bufferista. Vertex bufferit määrittää pisteet 3D-avaruudessa. Pisteillä pitää aina olla tieto, missä paikassa ne ovat, mutta niille voi myös antaa muunkinlaista tietoa, kuten väri tai tekstuurikoordinaatit. Index bufferit rakentaa kolmioita vertex bufferin avulla yhdistämällä kolme pistettä ja piirtämällä kolmion näiden välille. Tämän avulla myös kerrotaan näytönohjaimella, miten päin kyseinen kolmio on katsojaan nähden. Jos pisteiden järjestys on myötäpäivään, on kolmio katsojasta poispäin, kun taas jos järjestys on vastapäivään, on kolmio katsojaan päin. (Ogre3D 2011d.)

4.2 ManualObject

ManualObjectit ovat malleja, joiden luomiseen ei tarvitse mallintamisohjelmistoa. Täten ne eivät ole aivan sama asia kuin meshit, mutta ManualObjecteista voidaan kuitenkin luoda meshejä. Kaikki ManualObjectit luodaan ohjelmointikielen avulla.

```
ManualObject* manual = mSceneMgr->createManualObject("manual");
manual->begin("BaseWhiteNoLighting", RenderOpera-
tion::OT_LINE_STRIP);

manual->position(-100.0, -100.0, 0.0);
manual->position( 100.0, -100.0, 0.0);
manual->position( 100.0,  100.0, 0.0);
manual->position(-100.0,  100.0, 0.0);
manual->position(-100.0, -100.0, 0.0);

manual->end();
mSceneMgr->getRootSceneNode()->createChildSceneNode()-
>attachObject(manual);
```

KUVIO 1. ManualObjectin avulla luotu neliö (Ogre3D 2012.)

Edellä olleessa esimerkissä on luotu ManualObject, joka on neliö. Se on luotu

määrittämällä pelkkiä verteksejä, joten kyseisestä mallista ei voida luoda meshiä.

```
ManualObject* manual = mSceneMgr->createManualObject("manual");
manual->begin("BaseWhiteNoLighting", RenderOpera-
tion::OT_LINE_STRIP);

// määritellään verteksien paikat
manual->position(-100.0, -100.0, 0.0);
manual->position( 100.0, -100.0, 0.0);
manual->position( 100.0,  100.0, 0.0);
manual->position(-100.0,  100.0, 0.0);

// määritellään verteksien käyttö viittaamalla indekseihin
manual->index(0);
manual->index(1);
manual->index(2);
manual->index(3);
manual->index(0);

manual->end();
mSceneMgr->getRootSceneNode()->createChildSceneNode()-
>attachObject(manual);
```

KUVIO 2. ManualObjectin avulla luotu neliö, jossa on käytetty indeksejä (Ogre3D 2012.)

Kyseisessä ohjelmointiesimerkissä näkyy kuinka vertex ja index bufferit toimivat ohjelmoinnin näkökulmasta. Tällä esimerkillä saadaan aikaiseksi samanlainen ManualObject kuin ylemmässä esimerkissä eli neliö. Koska kyseiseen ManualObjectiin on määritetty myös indeksit, voidaan siitä luoda myös mesh. (Ogre3D 2012b.)

4.3 Materiaalit

Materiaalien avulla määritetään, miltä meshin pinta näyttää, kun se on renderöity. Yleisin materiaalien käyttötapa on määritellä tietty tekstuuri meshille. Tällöin saadaan valittu kuva piirtymään meshin pinnalle. Materiaaleilla voidaan myös antaa mesheille värejä ja läpinäkyvyyttä tai määrittää valaisukyky ja shaderit. Lähes kaikkia meshien ulkonäköön liittyviä ominaisuuksia säädetään materiaalien avulla. OGRE tukee muun muassa PNG-, JPEG-, TGA-, BMP-, HDR- ja cubemap-kuvatiedostoja. (Ogre3D 2011b.)

Materiaaleja voidaan määritellä OGRE:ssa materiaaliskripteillä tai ohjelmointikielellä suoraan koodiin. Materiaaliskriptejä käytettäessä luodaan .material-tekstitiedosto, johon kaikki kyseiseen materiaaliin vaikuttavat ominaisuudet kirja-

taan. Skriptien hyvä puoli on se, että ne ovat omissa tekstitiedostoissaan ja ne ladataan vain, kun niitä tarvitaan. Jos materiaalit määritellään suoraan ohjelman koodiin, joudutaan se lataamaan joka kerta, vaikka kyseistä materiaalia ei käytettäisi sillä hetkellä. (Ogre3D 2009c.)

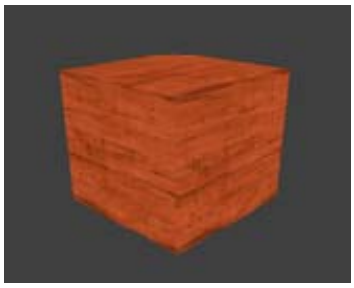
```
Ogre::MaterialPtr lMaterial = lMaterialManag-
er.create("M_Lighting+OneTexture",lNameOfResourceGroup);
Ogre::Technique* lFirstTechnique = lMaterial->getTechnique(0);
Ogre::Pass* lFirstPass = lFirstTechnique->getPass(0);

lFirstPass->setDiffuse(0.8f, 0.2f, 0.2f,1.0f);
lFirstPass->setAmbient(0.3f, 0.3f, 0.3f);
lFirstPass->setSpecular(1.0f, 1.0f, 1.0f, 1.0f);
lFirstPass->setShininess(64.0f);
lFirstPass->setSelfIllumination(0.5f, 0.5f, 0.5f);

Ogre::TextureUnitState* lTextureUnit = lFirstPass-
>createTextureUnitState();
lTextureUnit->setTextureName("puu.jpg", Ogre::TEX_TYPE_2D);
lTextureUnit->setTextureCoordSet(0);
```

KUVIO 3. Materiaalin määrittely ohjelmointikielellä (Ogre3D 2011.)

Yllä on tehty materiaalin määrittely suoraan sovellukseen ohjelmointikielen avulla. Sama määrittely voidaan myös hoitaa materiaaliskriptin avulla. Alla olevan skriptin avulla saadaan aikaiseksi täysin sama lopputulos kuin yläpuolella olevalla koodilla.

```
material M_Lighting+OneTexture
{
  technique
  {
    pass
    {
      lighting on
      ambient 0.3 0.3 0.3
      diffuse 0.8 0.2 0.2 1.0
      specular 1.0 1.0 1.0 1.0 64.0
      emissive 0.5 0.5 0.5 1.0
      texture_unit
      {
         texture puu.jpg 2d
         tex_coord_set 0
      }
    }
  }
}
```

KUVIO 4. Materiaalin määrittely materiaaliskriptin avulla (Ogre3D 2011.)



KUVA 2. Kuutio, johon on liitetty edellä oleva materiaaliskripti

4.4 Technique ja pass

Edellisen sivun materiaaliskriptistä voi nähdä kaksi sen pääosaa. Yhdessä skriptissä on mahdollista olla monta eri technique-osaa, joista vain yksi ajetaan tietyllä kokoonpanolla ja hetkellä. Techniquet voidaan tehdä erilaisten tietokoneiden varalle. Jos käyttäjällä on käytössään näytönohjain, jonka avulla hän pystyy käyttämään kaikkia mahdollisia sovelluksessa käytettyjä ominaisuuksia, valitsee sovellus techniquen, joka ei karsi ominaisuuksia. Jos taas käyttäjällä on näytönohjain, joka ei tue kaikkia ominaisuuksia, valitaan käyttäjälle technique, joka ottaa pois käytöstä ominaisuudet, joita hän ei voi käyttää. Techniquen avulla voidaan myös säätää materiaaliin Level Of Detail-asetuksia, jolloin kaukana oleviin esineisiin käytetään techniquea, joka ei vie paljoa laskentatehoa. (Ogre3D 2009e.)

Toinen skriptien pääosa on pass-osa. Pass tarkoittaa yhtä renderöintikertaa tietylle mallille. Yhdellä techniquella voi olla yhdestä kuuteentoista eri passia. Käyttämällä useampia passeja saadaan aikaan erilaisia tehosteita. Erilaisia attribuutteja passeille on OGRE:ssa lähes neljäkymmentä. Näihin kuuluu esimerkiksi ambientväri, diffuse-väri, emissiivisyyden määrä, varjostuksen tapa ja iteraatioiden eli toistojen määrä. (Ogre3D 2009d.)

5 SISÄLLÖN LUONTI

Vaikka OGRE:een pystyykin tekemään meshejä ja muuta sisältöä OGRE:n ohjelmointikielen avulla, on sisällön luonti paljon helpompaa, jos siihen käyttää mallintamisohjelmistoja, jotka on suunniteltu tämänkaltaisen sisällön tekoon. Kuten edellä mainittiin, on meshien luominen mahdollista myös ManualObjectin avulla, mutta vähänkin monimutkaisempien kappaleiden tekeminen on sitä käyttämällä vaikeaa ja aikaa vievää. Yksinkertaistenkin kappaleiden luonti on mallintaen paljon helpompaa kuin ohjelmoimalla, ainakin jos käyttäjä osaa mallintaa oikeilla mallintamisohjelmistoilla vähääkään.

Materiaaliskriptien avulla pystyy kappaleille luomaan monenlaisia ominaisuuksia, jotka voivat antaa ympäristölle lisää elävyyttä ja tunnelmallisuutta. Niiden tärkein ulkonäköön vaikuttava ominaisuus on kuitenkin tekstuurien lisääminen mesheihin. Tekstuurit täytyy valmistaa jossain toisessa ohjelmistossa, kuvata niitä itse tai hankkia valmiita tekstuureja esimerkiksi kuvapankeista. Myös monet muut korkeuskartat, kuten normal mapit, täytyy valmistaa muissa ohjelmistoissa tai hankkia valmiiksi tehtyinä.

5.1 Tarvittava ohjelmisto

Jotta OGRE:lle tarvittavia ohjelmointikieliä pystytään ohjelmoimaan ja kääntämään, tarvitaan jokin ohjelmankehitysympäristö. Jokaisella tuetulla käyttöjärjestelmällä on monia eri kehitysympäristöjä, joista on mahdollista valita se, jota halutaan käyttää OGRE-sovellusten tekemiseen.

Windows-käyttöjärjestelmille OGRE:n yhteisö tukee tällä hetkellä kolmea eri ohjelmankehityssovellusta: Microsoftin omaa Visual Studiota, Code::Blocksia ja Nokian QtCreatoria. Linux-käyttöjärjestelmillä on mahdollista käyttää myös Code:Blocksia ja QtCreatoria mutta myös muita sovelluksia, kuten Eclipsea, NetBeansia, Kdevelopia ja Autotoolsia. Mac OS X:ssä käyttäjä voi ohjelmoida Applen Xcode:lla. Käyttäjä voi käyttää kaikissa käyttöjärjestelmissä myös muita ohjelmankehityssovelluksia, kuin mitä OGRE:n yhteisö suosittelee. Tämä saattaa kuitenkin lisätä mahdollisten ongelmien määrä, eikä ongelmanratkaisu tule olemaan yhtä helppoa kuin yhteisön jo valmiiksi testatuilla ohjelmilla.

Monet ohjelmankehitysympäristöt vaativat muitakin erillisiä ohjelmia, että OGRE saadaan toimimaan niiden avulla. Esimerkiksi Code::Blocks vaatii MinGW-kehitysympäristön, jotta OGRE:n versio 1.7.2 saadaan käännettyä. Monissa ympäristöissä on myös mahdollista käyttää erillistä Cmake-rakennussovellusta. Kun Cmakea käytetään, tekee se ohjaustiedostoja valitulle kehityssovellukselle, kääntää lähdekoodin ja luo kirjastot. (Ogre3D 2011h.)

5.1.1 Mallintamisohjelmistot

Jotta OGRE:een voitaisiin saada 3D-objekteja, täytyy ne ensin luoda jokin muun sovelluksen avulla. Tähän tarvitaan jokin mallintamisohjelmisto, joita on tarjolla monia erilaisia. Kuitenkaan aivan kaikilla kyseisillä ohjelmistoilla ei OGRE-sovelluksiin pysty sisältöä luomaan. Kaikki mallintamisohjelmistot vaativat exporter-pluginin, joka muuttaa käyttäjän tekemät 3D-mallinnukset OGRE:n käyttämään .mesh-muotoon.

OGRE:n exporter-plugin on saatavilla ainakin kolmelletoista eri mallinnusohjelmistoille, kuten 3ds Maxille, Mayalle, Blenderille, LightWavelle ja Cinema 4D:lle. Plugineissa on erilaisia ominaisuuksia, ja senpä takia joillekin ohjelmistoille on luotu monta eri pluginia, jolla vienti on mahdollista tehdä. 3ds Maxillekin on olemassa yhdeksän eri pluginia, joissa kaikissa on hieman erilaiset ominaisuudet. Esimerkiksi 3DS2Mesh-plugin muuttaa .3ds-tiedostot suoraan .mesh- ja .material-tiedostoiksi, kun taas OgreMax Scene Exporterilla voi säätää paljon tarkemmin mitä kaikkea halutaan muuttaa ja malleja voi myös tarkastella mukana tulevalla ohjelmalla, ilman että niitä pitäisi aina mennä tarkastelemaan OGRE-sovelluksen puolelle. (Ogre3D 2012c.)

5.1.2 Kuvankäsittelyohjelmistot

Joskus tekstuureja, joita sovelluksessa käytetään, ei tarvitse ollenkaan muokata. Jos kuvatiedostot hankkii 3D-artisteille erikoistuneista kuvapankeista, on kuvat jo valmiiksi muokattu niin, että ne toimivat helposti 3D-objekteissa. Kuitenkin tekstuureja joutuu usein muokkaamaan ja luomaan kuvankäsittelyohjelmissa. Jos tekstuurit on myös kuvattu itse digitaalikameralla, pitää ne joka tapauksessa skaalata pienemmiksi, sillä reaaliaikaiset sovellukset, joita OGRE:n avulla ajetaan, ovat

liian raskaita nykypäivän tietokoneille, jos sovelluksissa on käytetty erittäin suuriresoluutioisia kuvia.

Suosittuja kuvankäsittelyyn soveltuvia ohjelmia ovat muun muassa Photoshop ja GIMP, joissa molemmissa on tarvittavia ominaisuuksia kuvien muokkaamiseen. GIMP ei ole aivan yhtä monipuolinen ja helppokäyttöinen kuin Photoshop, mutta se on saatavilla useammalle eri käyttöjärjestelmälle ja täysin ilmaiseksi.

Normal- ja bump-mappien ja muiden korkeuskarttojen tekemiseen voidaan käyttää myös monia mallinnusohjelmistoja, kuten 3ds Maxia ja Mayaa. Tämä yleensä onnistuu, kun mallista tehdään kaksi versiota, joista toisessa on vähemmän polygoneja kuin mallissa, jonka halutaan olevan lopullinen tulos. Tämän jälkeen vähäpolygoninen malli siirretään toisen päälle, niin että se peittää lopullisen tuloksen ja annetaan ohjelmiston laskea korkeuskartta kyseisten mallien avulla. Korkeuskarttoja voidaan luoda myös keinotekoisesti siihen tarkoitettujen ohjelmistojen avulla. Tämänkaltaisia ohjelmia on esimerkiksi CrazyBump, jonka avulla voidaan luoda melkein mistä tahansa kuvatiedostosta erilaisia korkeuskarttoja. Myös kuvankäsittelyohjelmiin, kuten Photoshopiin ja GIMP:iin on saatavilla plugineja, joiden avulla pystyy luomaan normal mappeja. Ne eivät yllä aivan yhtä hyvään laatuun kuin erityisesti tätä varten luodut ohjelmat, mutta ovat silti hyvä vaihtoehto, jos laadun ei tarvitse olla aivan paras mahdollinen.

5.2 Rajoitukset

Vaikka laitteiston vähimmäisvaatimukset riippuvat paljolti siitä, minkälaisen sovelluksen rakentaa ja kuinka paljon objekteja ja kuinka tarkasti mallinnettuja ne ovat, on OGRE:lla kuitenkin vähimmäisvaatimuksia esimerkiksi näytönohjainten suhteen. Muiden komponenttien kanssa vaatimuksia ei ole muuten kuin sen perusteella, kuinka paljon laskentatehoa mallien ja sovelluksen sulava pyöritys vaatii.

Nykyinen OGRE:n versio tukee DirectX:ää yhdeksännestä versiosta alkaen, joka sekin on jo vuodelta 2002. Ensimmäiset näytönohjaimet, jotka tukivat DirectX 9:ää, olivat Nvidian Geforce FX -sarja ja ATI:n R300-sarja. OpenGL:n osalta OGRE tukee kuitenkin jopa tätäkin vanhempia näytönohjaimia. Jopa Geforce 2 tai Radeon 7500 pystyisivät ajamaan OGRE:a, vaikkakin suositellut näytönohjaimet

ovat Geforce 4 ja Radeon 9600. (Ogre3D 2010b.)

Kaikki kyseiset näytönohjaimet ovat nykyisiin näytönohjaimiin verrattuna erittäin heikkotehoisia, joten nykyisillä tietokoneilla ei pitäisi olla suuria ongelmia perus OGRE-sovelluksen ajamiseen. Vaikka OGRE:n yhteisö ei suoranaisesti tuekaan Intelin integroituja näytönohjaimia, ei niiden kanssa suurempia ongelmia pitäisi enää ilmetä, sillä myös ne ovat tehokkaampia kuin kymmenen vuotta vanhat erillisnäytönohjaimet.

Tiukimmat laitteistovaatimukset tulevat vastaan ennemminkin sovelluksissa, joiden avulla sisältöä tehdään kuin OGRE-sovelluksella, jota ollaan tekemässä. Tämänlaisia ohjelmia ovat muun muassa 3ds Max ja Photoshop, joiden vaatimukset kasvavat vuosi vuodelta. Sisällönluominen saattaa myös käydä erittäin hitaaksi ja työlääksi, jos laitteisto ei ole tarpeeksi tehokas suorittamaan näitä ohjelmia tarpeeksi sulavasti.

6 TYÖKALUT

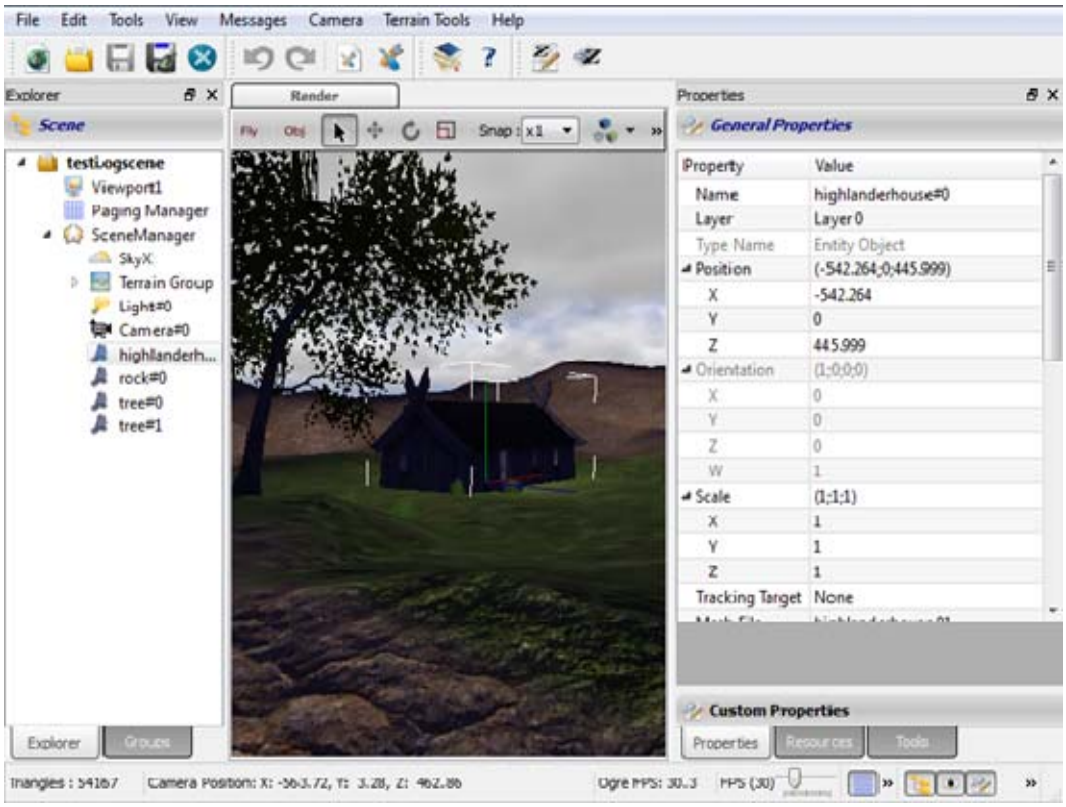
Koska OGRE:ssa itsessään ei ole sisäänrakennettuja työkaluja, vaan kaikki materiaalit ja mallit tuodaan OGRE:een ja ohjelmointikielen avulla tehdään halutut asiat, ovat erilaiset työkalut erittäin hyödyllisiä OGRE:n käyttäjälle.

OGRE:n käyttöön on olemassa työkaluja, joiden avulla OGRE:n käyttö helpottuu tietyissä tilanteissa merkittävästi. Näitä työkaluja on tehty moneen eri tarkoitukseen, ja kaikki niistä ei ole suoranaisesti sisällönluomiseen, mutta silti ne voivat helpottaa OGRE:n käyttämistä.

Ogitor

Mallien sijoittelu ja halutunlaisen näkymän aikaansaaminen voi olla pelkästään ohjelmoimalla erittäin hankalaa ja turhauttavaa, sillä ohjelmoinnin jälkeen täytyy sovellus aina kääntää uudestaan, jotta pystytään näkemään muutokset, joita sovellukseen on tehty. Tätä helpottamaan on tehty editoreja, joiden avulla nähdään suoraan, miltä ympäristö milläkin hetkellä näyttää. Yksi näistä editoreista on Ogitor, Ogre editor, joka on niin kutsuttu WYSIWYG-editori. Kyseinen lyhenne tulee sanoista What You See Is What You Get, joka tarkoittaa sitä, että luomalla jonkinlaisen ympäristön Ogitorissa, tulee ympäristö näyttämään samalta myös lopullisessa OGRE-sovelluksessa. (Moen 2011.)

Ogitor on plugin-pohjainen, joten siihen on mahdollista liittää monia ominaisuuksia, joita alkuperäinen ohjelma ei sisällä. Yleisimpiä ja ehkäpä hyödyllisimpiä plugineita ovat monet erilaisia tilanteita simuloivat pluginit, kuten vedenalaista maailmaa tai vuorokauden vaihteluita säätelevät liitännäiset. Myös käyttöliittymää on mahdollista muokata Angelscriptin avulla, sen näköiseksi kun sen itse haluaa. Ogitoria on mahdollista käyttää Windows-, Mac OS X- ja Linux -käyttöjärjestelmillä, joten sen käyttö on mahdollista kaikilla käyttöjärjestelmillä, joilla myös OGRE:a on mahdollista käyttää. (Moen 2011.)



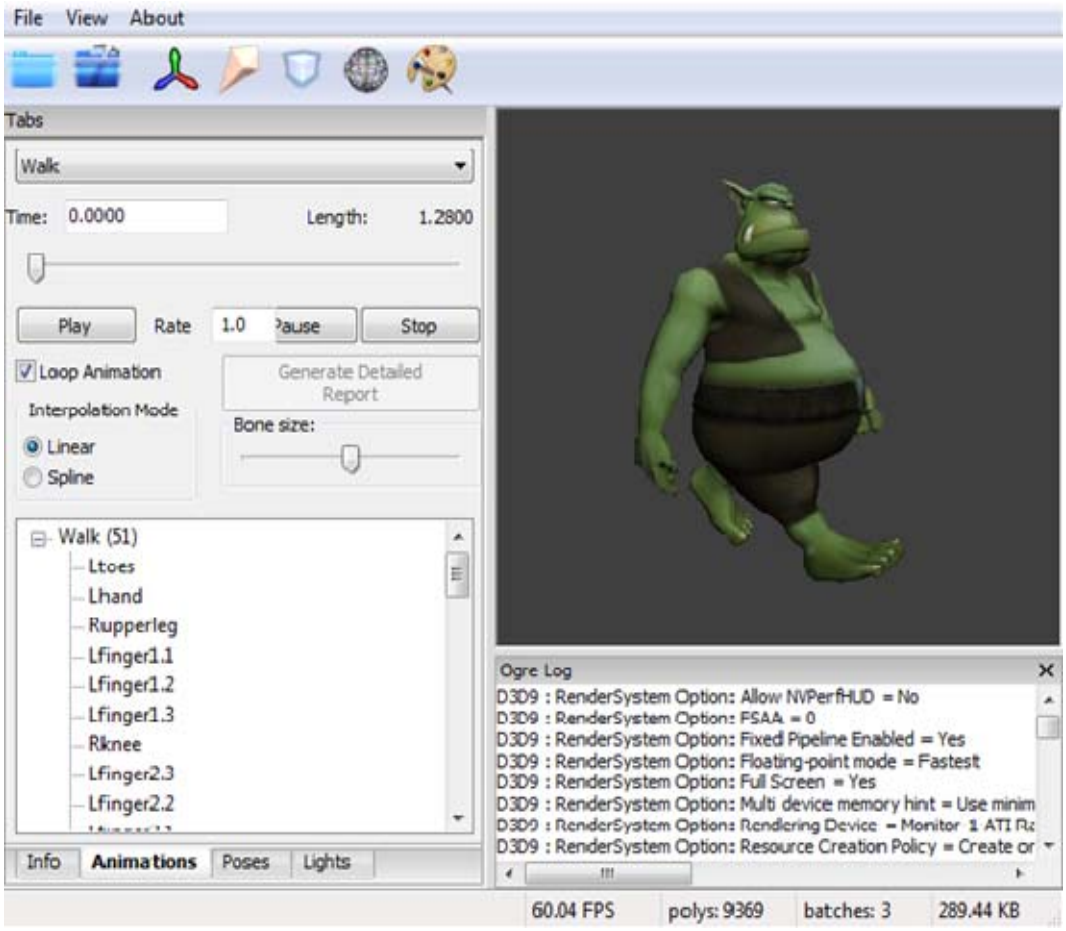
KUVA 3. Ogitorin käyttöliittymä

Ogre Meshy

Ogre Meshy on työkalu, jonka avulla pystytään tarkastelemaan meshejä ilman niiden viemistä itse sovellukseen tai erilliseen editoriin. Jotkin exporterit myös sisältävät kyseisen toiminnon, joten Ogre Meshystä ei ole niin suurta hyötyä niiden exportereiden käyttäjille, mutta heille, ketkä eivät pysty tämänkaltaisia exportereja käyttämään, on Ogre Meshy erittäin hyödyllinen työkalu.

Ogre Meshyllä pääasiallisen käyttötavan eli meshien tarkastelun lisäksi sillä pystytään näkemään tietoja meshistä, materiaalista ja mahdollisista animaatioista. Normaaleja animaatioita ja hahmoanimaatioita pystyy myös ajamaan Ogre Meshyllä ja myös animaation keston skaalaus on mahdollista. (Ogre3D 2011e.)

Ogre Meshyllä pystyy lukemaan vain OGRE:n .mesh-tiedostoja. Se ei ole tällä hetkellä saatavilla Mac OS X:lle vaan pelkästään Windows- ja Linux -käyttöjärjestelmille. (Ogre3D 2011e.)



KUVA 4. Ogre Meshyn käyttöliittymä

Particle Universe

Partikkeliefektien avulla ovat kaikenlaiset efektit näyttävämpiä ja vähemmän laskentatehoa vieviä kuin monilla muilla mahdollisilla tavoilla. Tästä syystä partikkelieditorit ovat erittäin hyödyllisiä, kun halutaan saada aikaiseksi näyttävä kokonaisuus lopullisessa sovelluksessa.

Particle Universe on maksullinen ohjelma, jonka avulla pystytään tekemään partikkeliefektejä, kuten savua, tulta, räjähdyksiä ja sadetta, OGRE-sovelluksiin. Se sisältää editorin, jossa efektit tehdään puujärjestelmän avulla. Editorissa pystyy myös näkemään reaaliajassa, miltä partikkeliefekti tulee näyttämään sovelluksen puolella. Saatavilla on myös maksullinen partikkelikokoelma, joka sisältää yli 120 erilaista valmista partikkeliefektiä, jolloin käyttäjän ei tarvitse kuin valita hänen sovellukseen sopivat efektit. Particle Universe on saatavilla ainoastaan Windows-käyttöjärjestelmille. (FXpression 2012.)

Caelum

Caelum on plugin OGRE:lle, jolla on mahdollista luoda ja muunnella tunnelmallisia ympäristötehosteita, kuten taivasta ja sen väriä, pilviä ja erilaisia sääilmiöitä. Caelum on myös mahdollista liittää Ogitoriin, jolloin sen käyttö on helpompaa ja täten se myös mahdollistaa hienompien sääilmiöiden luonnin Ogitorissa. Caelum on saatavilla Windows ja Linux-käyttöjärjestelmille. Vaikka Ogitor on saatavilla myös Mac OS X:lle, valitettavasti Caelum ei ole. (Ogre3D 2011a.)

Scenemanagerit

OGRE:lle on olemassa monia erilaisia ”scenemanagereja”. Scenet eli ympäristöt voivat sisältää staattista geometriaa, valoja, kameroita ja lähes kaiken mikä vaikuttaa näkyvällä tavalla sovelluksiin. Scenemanagerien avulla saadaan kaikki tämä optimoitua ja koottua kasaan sovellusta varten. Erilaisia scenejä varten voi käyttää eri manageria, kuten esimerkiksi eri managerit ulkotilalle ja sisätilalle. Eri managereissa on erilaisia ominaisuuksia riippuen mihin tarkoitukseen ne ovat tehty. (Ogre3D 2011g.)

OgreXmlConverter

OgreXmlConverterin avulla saadaan .mesh ja .skeleton-tiedostot muutettua xml-tiedostoiksi ja päinvastoin. Näin myös mallien data tarkastelu ja muuntelu onnistuvat exporttauksen jälkeenkin. (Ogre3D 2011f.)

Monet muuntimet muuntavat mallit xml-muotoon, joten OgreXmlConverteria vaaditaan, jotta mallit saadaan OGRE:n vaatimaan .mesh-muotoon. Kyseinen muunnin sisältyy myös OgreCommandLineTools-pakettiin, jossa kaikki paketissa olevat työkalut nimensä mukaisesti ajetaan komentokehotteesta. (Ogre3D 2011f.)

Ogre Application Wizard

Ennen kuin OGRE-sovelluksia voi alkaa ohjelmoida, täytyvät tarvittavat ohjelmankehitysympäristöt asentaa ja OGRE-kohtaiset asetukset valmistella, jotta OGRE-sovelluksia pystytään kääntämään. Vaikka OGRE-yhteisö onkin kirjoittanut perusteellisen ohjeistuksen näiden valmistelujen tekemiseen, voi se olla kuitenkin monelle aloittelijalle ja asiaan perehtymättömälle liian hankalaa.

Ogre Application Wizard on luotu sen takia, ettei käyttäjän tarvitse tehdä kaikkia valmisteluja, vaan ne on mahdollista hoitaa ajamalla kyseinen sovellus. Ogre Application Wizard on saatavilla Visual Studio, KDevelop ja Code::Blocks-kehityssovelluksille. (Atlassian 2012.)

7 OGRE:A KÄYTTÄVÄT PELIMOOTTORIT JA PELIT

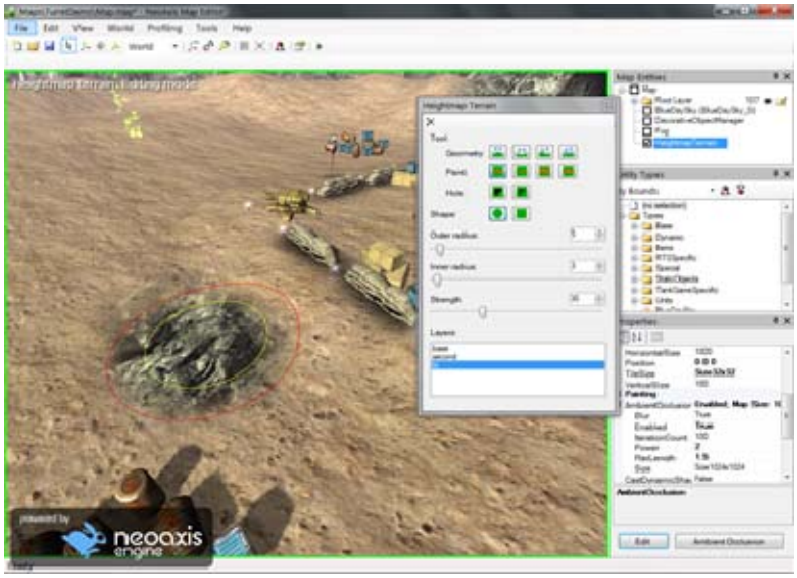
OGRE:a on käytetty monissa eri projekteissa, kuten peleissä ja pelimoottoreissa. Monet niistä ovat kuitenkin ilmaisia ja eivät ole tulleet kovinkaan tunnetuiksi suurelle yleisölle. Koska OGRE on itsessään ilmainen, monet tahot ovat käyttäneet sitä myös pienempiin projekteihinsa, jottei heidän tarvitse rakentaa koko sovellusta alusta asti tai maksaa joistain kaupallisista vaihtoehdoista.

OGRE:a on kuitenkin käytetty myös joissain erittäin suosituissakin projekteissa. Tämä on lisännyt sen näkyvyyttä ja auttanut muitakin pelinkehittäjiä löytämään OGRE:n omiin projekteihinsa.

NeoAxis Game Engine

NeoAxis Game Engine on yksi suosituimmista OGRE:en perustuvista pelimoottoreista. Se sisältää siis OGRE:n grafiikkamoottorin ja on lisännyt sen ympärille muita pelimoottoreille ominaisia ominaisuuksia. (Ogre3D 2012d.)

Toisin kuin OGRE, NeoAxis sisältää myös editorin, joka helpottaa pelien visuaalista suunnittelua. NeoAxis on myös lisännyt joitakin graafisia ominaisuuksia, joilla sovelluksista saadaan näyttävämpiä ja helpommin käytettäviä. Yksi merkittävistä uudistuksista on tuki COLLADA-tiedostoille, jolloin export-plugineita ei välttämättä tarvitse. (NeoAxis Group 2012.)



KUVA 5. NeoAxis Game Enginen käyttöliittymä (NeoAxis Group 2011.)

Torchlight

Torchlight on eräs parhaiten menestyneistä peleistä, joka on tehty käyttäen OGRE:a. Runic Games kehitti sen vuonna 2009 Windowsille ja myöhemmin myös Mac OS X:lle ja Xbox 360:lle. Torchlight on toimintaroolipeli, jossa pelaaja ohjaa hahmoaan läpi luolastojen suorittaen tehtäviä. Heinäkuussa 2011 kehittäjät ilmoittivat myyneensä Torchlightia yli miljoonan kappaleen verran. (Orland 2011.)

KUVA 6. Torchlight-peli (Runic Games 2012.)

Runic Games julkaisi myös TorchED-editorin, jonka avulla kaikki käyttäjät voivat tehdä peliin omia kenttiään. Pelistä on tulossa myös jatko-osa Torchlight 2, jonka on tarkoitus sisältää myös moninpeli, jota alkuperäisessä pelissä ei ollut. (Runic Games 2009, Runic Games 2011.)

8 CASE

8.1 Tavoitteet

Case-projektissa on tarkoituksena luoda sovellus, joka toimii esittely-ympäristönä erilaisille mediatekniikan ja mahdollisesti myös muiden alojen töille. Tavoitteena on saada aikaiseksi helppokäyttöinen käyttöliittymä ja visuaalisesti näyttävä ulkoasu.

Sovellus luodaan OGRE:n avulla ja erilaisia työkaluja ja plugineja käytetään helpottamaan ja nopeuttamaan prosessia. Sisältö tullaan luomaan suurimmaksi osaksi 3ds Max -mallinnusohjelmistolla. Sovellus pyritään kuitenkin saamaan mahdollisimman kevyeksi, jotta myös vanhemmat tietokoneet pystyisivät sitä käyttämään.

Tavoitteena olisi myös saada sovellukseen jonkinlainen toiminto, jolla esitettävää materiaalia saisi helposti lisättyä esitettäväksi. Ennen projektin aloittamista ei kuitenkaan tietoa ole, kuinka helppoa kyseinen toiminto olisi saada aikaiseksi tai onko se ylipäätään mahdollista toteuttaa. Jos toimintoa ei saada sovellukseen, täytyisi sovellus luultavasti kääntää joka kerta, kun sisältöä siihen lisätään.

8.2 Suunnittelu

Koska ilman kunnollista suunnittelua on kaikenlaisten projektien toteuttaminen vaikeampaa kuin hyvin suunnitellun projektin, on myös yksi casen vaiheista lopullisen tuotteen suunnittelu. Casen tavoitteena on luoda sovelluksesta visuaalisesti näyttävä, mutta myös kevyt, joten alusta alkaen sovellus täytyy yrittää pitää mahdollisimman keveänä. Kuitenkin myös visuaalinen näyttävyys täytyy säilyttää, joten tämän takia sovellukselle täytyy saada aikaiseksi tasapaino visuaalisen näyttävyyden ja keveyden välille.

Koska erittäin korkealaatuisten 3D-mallien reaaliaikainen esittäminen sulavasti on lähes mahdotonta, täytyisi malleja ensin yksinkertaistaa tai muuttaa siten, ettei malli sisältäisi liikaa polygoneja tai suuriresoluutioisia tekstuureja. Kuitenkin jotta mahdollisimman korkealaatuisia 3D-malleja voidaan esittää, ei ympäristöön voida tehdä suuria määriä ylimääräisiä 3D-malleja tai muuta paljon resursseja kuluttavia

ominaisuuksia.

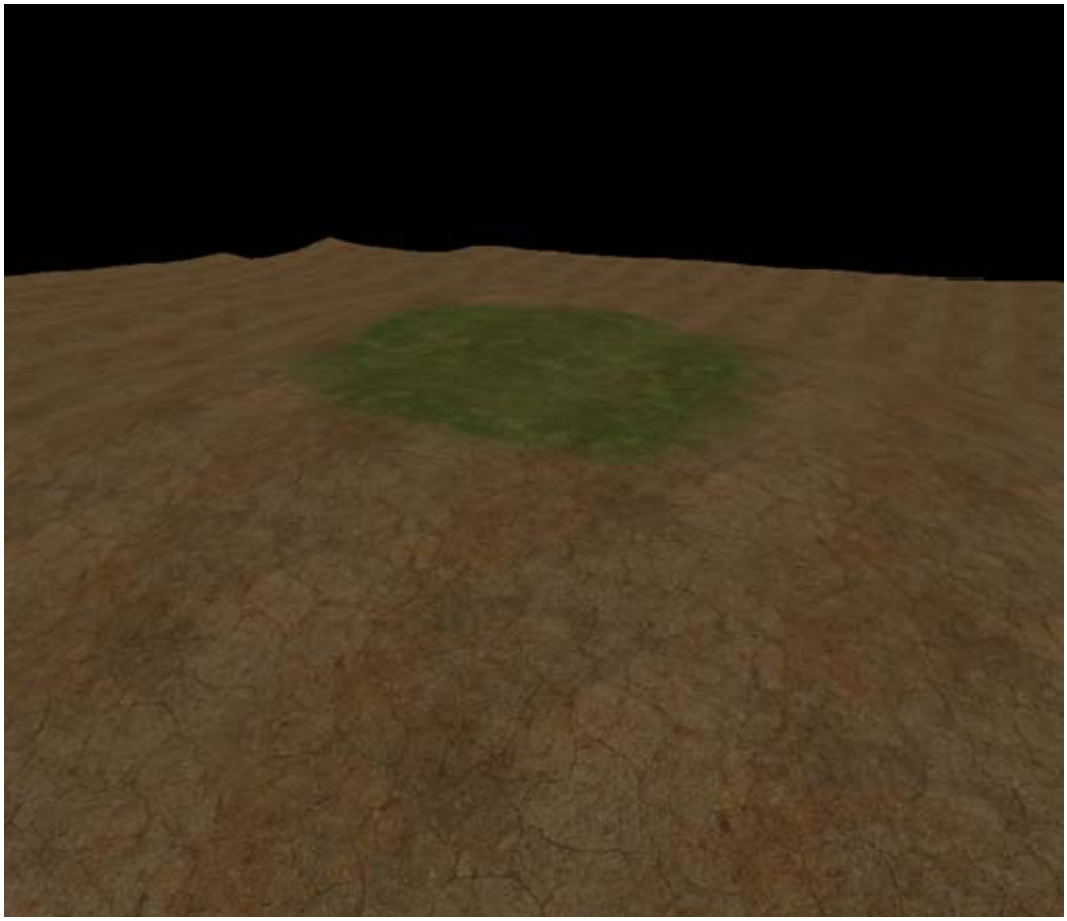
Myös kaksi tai useampi erilaista ympäristöä olisi hyödyksi, jotta sovellus olisi sulava kaikenlaisilla tietokoneilla. Yksi ympäristö voisi olla suunniteltu vain sovelluksen sulavuutta ajatellen, kun muut olisivat enemmän visuaalisuuteen painottavia. Tämä olisi myös monien mallien tai muiden esitettävien asioiden kannalta hyödyllistä, sillä kaikki niistä tuskin sopii samaan ympäristöön yhtä hyvin. Tämän takia jos päädytään tekemään vain yksi ympäristö, on tärkeää, että se tulisi olemaan mahdollisimman neutraali.

8.3 Ympäristön luonti

Ympäristön luonti päätettiin aloittaa jo aiemmin luvussa 6 esitellyn Ogitorin avulla. Sen avulla saadaan luotua kohtalaisen helposti lähtökohta sovelluksen ulkonäölle, jota voitaisiin myöhemmin muokata halutunlaiseksi.

Koska sovelluksen ulkonäölle ei ole mitään ennalta määriteltyjä vaatimuksia, suunnitellaan maisema täysin oman näkemyksen mukaan. Tästä syystä ympäristöksi valitaan pieni saari, joka toimii tietynlaisena esiintymislavana esitettävälle materiaalille.

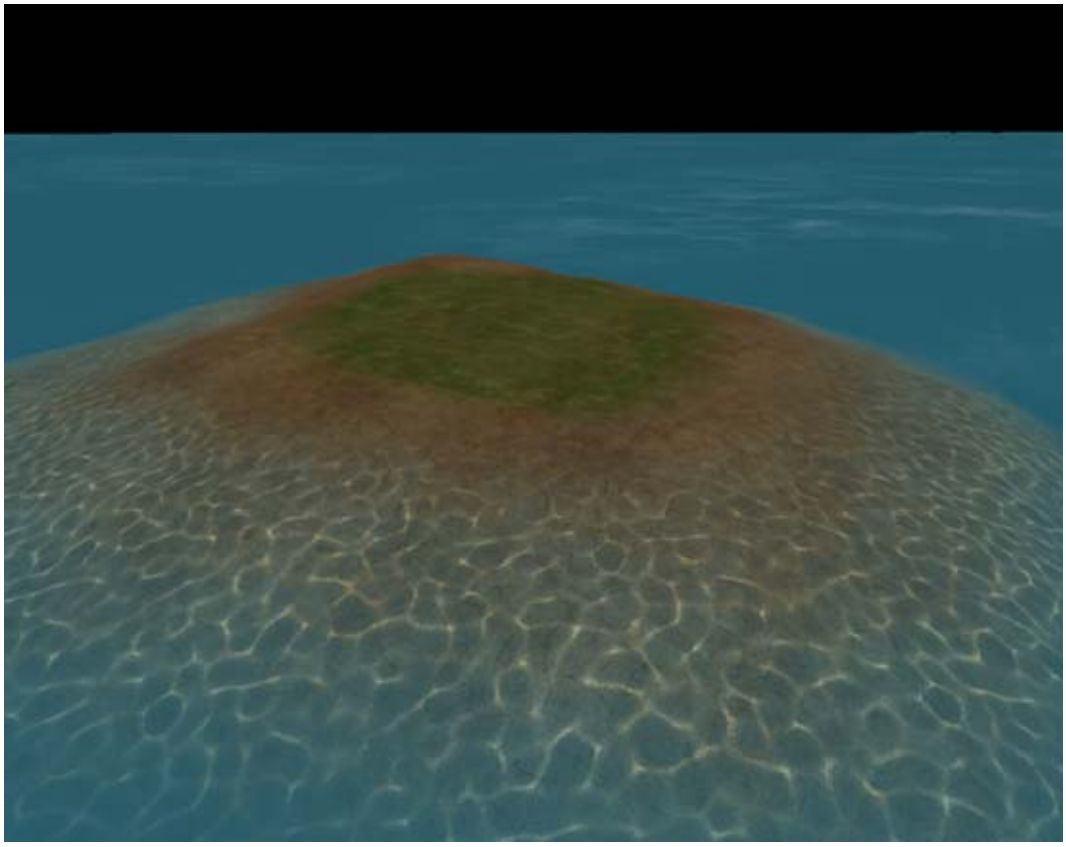
Ensimmäinen luonnos on kohtuullisen yksinkertainen ja sisältää vain maaston, josta saari muodostuu, kuten seuraavalla sivulla olevasta kuvasta 7 voidaan nähdä. Kuitenkin kyseisessä maastossakin on yli 70000 kolmiota, joista 3D-mallit koostuvat. Tämä määrä ei ole liian suuri yksinään tietokoneille, mutta saattaa koitua ongelmaksi, kun ympäristöön lisätään suuria määriä muitakin elementtejä. Kyseinen osa on kuitenkin vain yksi monista, joista ympäristön on tarkoitus koostua. Tämän takia toinen ympäristö voisi sisältää vain yksinkertainen tason, joka sisältäisi vain muutaman 3D-malleihin tarvittavan kolmion.



KUVA 7. Yksinkertainen maasto, josta pieni saari on tarkoitus luoda

Jotta ympäristö saadaan näyttämään muultakin kuin pelkältä autiomaalta, lisätään saaren ympärille myös vettä. Tässä vaiheessa projektia ei vielä ole tietoa, kuinka paljon resursseja tietokoneelta vesi vaatii, mutta kyseiseen ympäristöön se on lähes pakollinen elementti, sillä ilman vettä kyseinen maasto ei tule olemaan saari. Koska vedestä halutaan mahdollisimman näyttävän näköistä, käytetään sen tekemiseen Ogitorissakin toimivaa Hydrax-liitännäistä, jonka avulla vedestä saadaan realistisemman näköistä.

Seuraavan sivun kuvasta 8 näkyy kuinka vesi saa maiseman näyttämään hieman enemmän saarelta kuten tarkoitus oli. Kyseisestä maisemasta kuitenkin puuttuu vielä taivas kokonaan ja siinä ei myöskään ole minkäänlaista valaistusta. Kuvassa näkyy myös saaren rantaan osuvat heijastukset veden läpi, joita Hydrax:lla pystyy lisäämään. Tämän avulla vesi saadaan näyttämään realistisemmalta lähellä saarta.



KUVA 8. Saari ympäröity vedellä, jonka on tarkoitus lisätä visuaalista näyttävyyttä

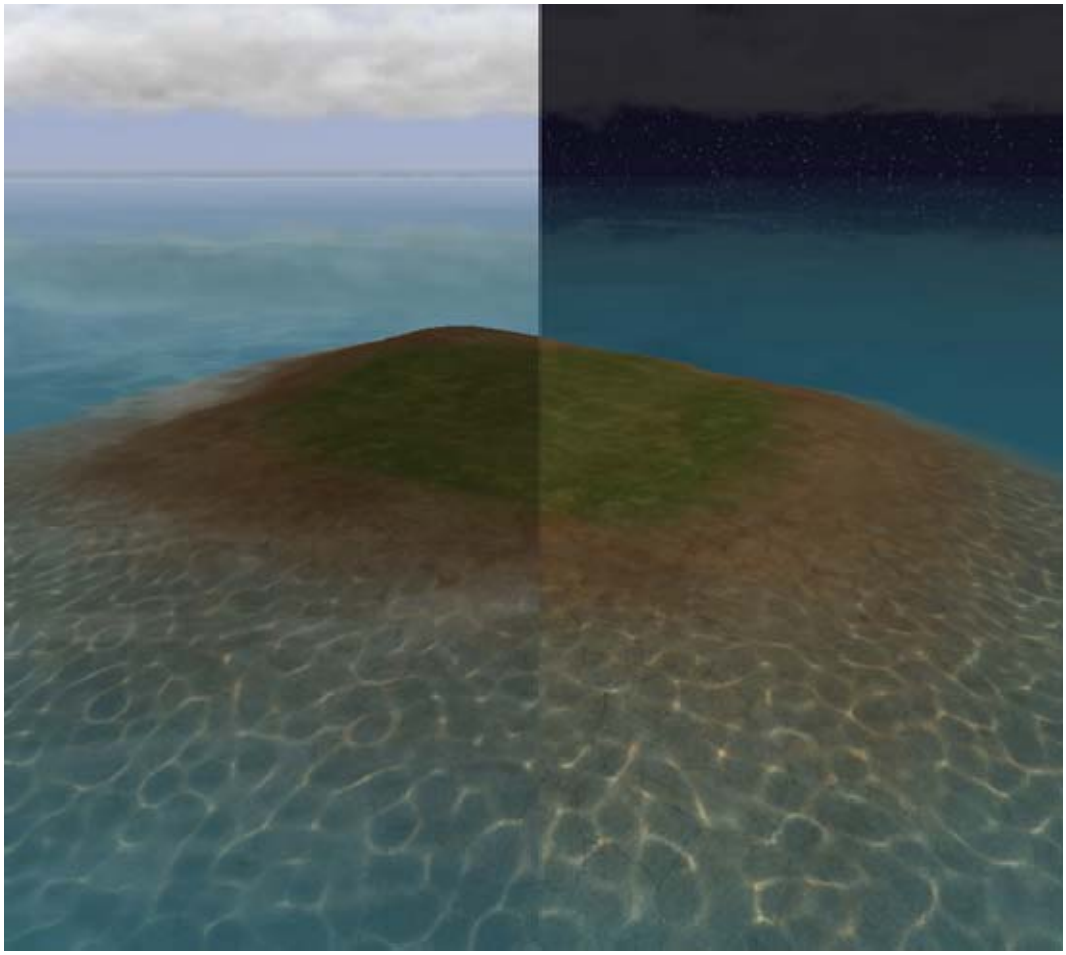
Tunnelmaa luomaan tarvitaan myös taivas, jonka avulla ympäristö voi alkaa jo muistuttaa enemmän oikeaa ympäristöä. Kun taivas lisätään sovellukseen tiettyjen liitännäisten avulla, lisää se ympäristöön myös valonlähteen. Myös muita valonlähteitä on mahdollista käyttää, mutta niistä ei ole paljoa hyötyä, jos sovelluksessa oletetaan auringon paistavan. Kuitenkin sovellukseen olisi mahdollista tehdä myös kaksi eri vuorokauden aikaa, joista toisessa paistaisi aurinko ja toisessa olisi yö. Tällöin myös muunlaisista valonlähteistä olisi hyötyä.

Taivasta koetettiin saada toimimaan ensin Caelum-liitännäisen avulla, joka myös esiteltiin luvussa 7. Jostain syystä ainakin koneella, jolla sovellusta tehtiin, koettiin kyseisen liitännäisen kanssa graafisia ongelmia ja pilvet eivät suostuneet näkymään oikealla tavalla vaan olivat suurimmaksi osaksi läpinäkyviä. Saattaa olla, ettei kyseistä ongelmaa olisi muilla tietokoneilla, mutta valitettavasti tätä ei pystytty testaamaan, vaan taivaan tekoa varten valittiin eri menetelmä.

KUVA 9. Caelum-liitännäisen aiheuttamat graafiset ongelmat

Koska Caelumin avulla taivasta ei saatu toimimaan halutulla tavalla, valittiin taivaan tekoon SkyX. Molemmat liitännäiset on tarkoitettu saman asian luomiseen, ja vaikka ominaisuudet ovatkin hieman erilaiset, pitäisi molemmilla pystyä tekemään yhtä näyttäviä kokonaisuuksia.

Kuvassa 10 on SkyX-liitännäisen avulla tehty ympäristöön taivas. Kyseisellä liitännäisellä saadaan aikaan myös vuorokaudenvaihteluita. Niiden avulla voidaan tehdä kaksi erillistä ympäristöä, joissa toisessa on päivä ja toisessa yö. Mahdollista olisi myös tehdä vain yksi ympäristö, missä päivä vaihtuu yöksi, mutta näin tehdessä katsojat keskittyisivät luultavasti enemmän nopeaan vuorokauden vaihteluun kuin itse esitettävään kohteeseen.

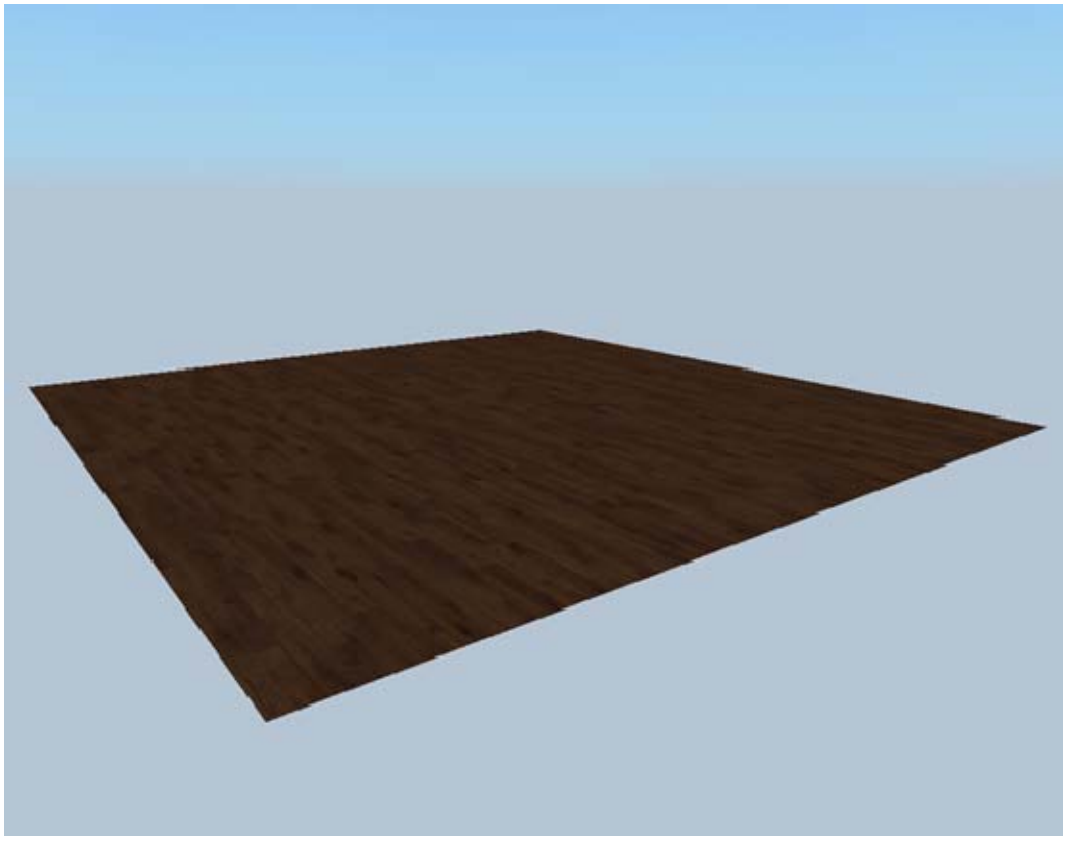


KUVA 10. Vertailu päivän ja yön välillä SkyX-liitännäistä käyttäessä

8.4 Vaihtoehtoinen ympäristö

Koska ensisijaiseen ympäristöön laitettiin ominaisuuksia, jotka vanhemmilla tietokoneilla saattaisi viedä liikaa resursseja, eikä täyttä varmuutta kaikkien ominaisuuksien toimivuudesta ole, päätettiin luoda vaihtoehtoinen ympäristö. Sen pitäisi olla yksinkertainen ja toimia mahdollisimman sulavasti. Sen takia alustaksi päätettiin laittaa pelkkä taso, johon on lisätty yksinkertainen tekstuuri.

Myös taustaksi valittiin yksinkertainen taivas, jottei resursseja kuluisi liikaa. Nämä kaksi elementtiä yhdistämällä vaihtoehtoisesta ympäristöstä ei tullut kovinkaan ihmeellisen näköinen, mutta sen käyttö onkin suurimmilta osin pelkästään tarkoitettu tietokoneille, jotka eivät pysty käyttämään ensisijaista ympäristöä.



KUVA 11. Vaihtoehtoinen näkymä sovelluksesta heikkotehoisemmille tietokoneille

8.5 Käyttöliittymä

Vaikka OGRE:een on saatavilla monia käyttöliittymäliitännäisiä, kuten CEGUI ja MyGUI, tulee projektin käyttöliittymä olemaan niin yksinkertainen, ettei monimutkaisempia liitännäisiä tarvita. Käyttöliittymä tulee rakentumaan vain muutamasta painikkeesta, joiden avulla voidaan esimerkiksi vaihtaa ympäristöä.

Sovelluksen kamera tulee osoittamaan todennäköisesti koko ajan samaan pisteeseen, ja kameraa tullaan liikuttamaan klikkaamalla hiirellä mitä tahansa pistettä näytöllä ja raahaamalla sitä haluttuun suuntaan. Tämän takia minkäänlaisia painikkeita liikkumiselle tai kameran pyörittämiselle ei tarvita.

Käyttöliittymän painikkeet yritetään saada kohtuullisen neutraaleiksi ja huomaamattomiksi, ettei käyttäjä häiriinny painikkeista tai ala tutkimaan kyseisiä painikkeita liikaa. Painikkeita katsomalla pitäisi kuitenkin olla täysin selvää, mitä ne tekevät eikä jättää mitään arvailujen varaan. Yhteensä painikkeita tulee olemaan viisi. Yksi painike sovelluksen lopettamiselle, kaksi painiketta ympäristön muuttamiselle ja kaksi painiketta esitettävän kohteen vaihtamiselle seuraavaan tai edel-

liseen kohteeseen.

KUVA 12. Painikkeet joita sovelluksessa voi klikkailla hiirellä

8.6 Lopputulos

Käyttöliittymän kanssa lopullinen sovellus näyttää jo hieman enemmän pelimäiseltä. Vaikka sovelluksessa ei kovin montaa ominaisuutta olekaan ja muutamia piti myös karsia pois, on lopputulos kuitenkin kohtalaisen toimiva sovellus, jota voi käyttää mallien ja muun materiaalin esittelyyn. Kuvasta 13 voi nähdä, miltä lopullinen sovellus näyttää, kun malli on ladattu saaren päälle.

KUVA 13. Ogre-malli valmiissa sovelluksessa

Sovellus kuitenkin osoittaa, että OGRE:lla pystyy tekemään visuaalisia sovelluksia samalla tavalla kuin muilla suositummilla pelimoottoreilla. Vaikka OGRE ei pystykään tekemään muuta kuin esittämään grafiikkaa, pystytään siihen myös lisäämään muita ominaisuuksia, joiden avulla sillä voidaan esimerkiksi kehittää pelejä ja muita interaktiivisia sovelluksia.

9 YHTEENVETO

Opinnäytetyössä käytiin läpi OGRE:n ominaisuuksia ja vertailtiin sitä muihin pelimoottoreihin. Käytännössä suurimmat eroavaisuudet muihin moottoreihin verrattuna ovat OGRE:n ilmaisuus, se on käytännössä pelkkä grafiikka-moottori ja editorin puuttuminen OGRE:sta ilman ulkopuolisia työkaluja. OGRE on käytännössä tarkoitettu henkilöille, jotka tietävät, mitä haluavat, eivätkä halua, että heidän täytyy käyttää tiettyjä pelimoottoreihin sidottuja työkaluja vaan saavat valita omansa.

Case-osiossa tehtiin OGRE:n avulla sovellus, jolla pystyy esittämään 3D-malleja ja muuta materiaalia. Osiossa selvisi, että myös OGRE:lla pystytään tekemään interaktiivisia visuaalisia sovelluksia, vaikka OGRE ei itsessään sisällä kaikkia tarvittavia ominaisuuksia. Sovellusten visuaalinen näyttävyys riippuu suurimmilta osin tekijän taidoista, joten lähes millä tahansa pelimoottorilla pystytään tekemään visuaalisesti näyttäviä esityksiä. Myös tekninen puoli riippuu myös paljon tekijästä, mutta kaikkia teknisiä rajoituksia ei voida kiertää, joten pelimoottorien tekniset rajoitukset ovat lähes ainoa este tekijöille.

Pelimoottorit kehittyvät koko ajan eteenpäin ja uudet versiot tuovat aina uusia ominaisuuksia. Näiden ominaisuuksien avulla pystytään sovelluksista tekemään yhä kehittyneempiä ja visuaalisempia. Osassa pelimoottoreista kuten OGRE:ssa on panostettu enemmän graafisiin ominaisuuksiin ja pelimoottorin laajennettavuuteen, kun taas joissain moottoreissa ei ole laajennusmahdollisuuksia ja suurin osa ominaisuuksista on käytettävissä ilman liitännäisiä.

LÄHTEET

Painetut lähteet

Junker, G. 2006. Pro OGRE 3D Programming. Berkeley: Apress.

Elektroniset lähteet

Atlassian. 2012. ogreappwizards [viitattu 29.3.2012]. Saatavissa: https://bitbucket.org/jacmoe/ogreappwizards

Crytek. 2012. CryENGINE Free Use [viitattu 14.3.2012]. Saatavissa: http://mycryengine.com/index.php?conid=70

Develop. 2011. Crytek: We're bringing our tech to Wii U [viitattu 14.3.2012]. Saatavissa: http://www.develop-online.net/news/38014/Crytek-Were-bringing-our-tech-to-Wii-U

DevMaster. 2011a. CryEngine 3 [viitattu 14.3.2012]. Saatavissa: http://devmaster.net/devdb/engines/cryengine-3

DevMaster. 2011b. Unity [viitattu 23.3.2012]. Saatavissa: http://devmaster.net/devdb/engines/unity

Epic Games. 2010. Licensing [viitattu 14.3.2012]. Saatavissa: http://udk.com/licensing

Epic Games. 2012. Platforms [viitattu 14.3.2012]. Saatavissa: http://www.unrealengine.com/platforms

Free Software Foundation. 2011. Why you shouldn't use the Lesser GPL for your next library [viitattu 22.3.2012]. Saatavissa: http://www.gnu.org/licenses/why-not-lgpl.html

FXpression. 2012. Express yourself with FXPression [viitattu 16.3.2012]. Saatavissa: http://www.fxpression.com/

Moen, J. 2011. HomePage [viitattu 15.3.2012]. Saatavissa: http://www.ogitor.org/HomePage

NeoAxis Group. 2012. Full-Featured Toolset [viitattu 1.4.2012]. Saatavissa: http://www.neoaxis.com/neoaxis/features/tools

Ogre3D. 2009a. Hardware Index Buffers [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/docs/manual/manual_60.html#SEC288

Ogre3D. 2009b. Hardware Vertex Buffers [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/docs/manual/manual_55.html#SEC280

Ogre3D. 2009c. Material Scripts [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/docs/manual/manual_14.html#SEC23

Ogre3D. 2009d. Passes [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/docs/manual/manual_16.html#SEC39

Ogre3D. 2009e. Techniques [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/docs/manual/manual_15.html#SEC33

Ogre3D. 2010a. Brief history of OGRE [viitattu 14.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/Brief+history+of+OGRE

Ogre3D. 2010b. Hardware [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/Hardware

Ogre3D. 2011a. Caelum [viitattu 23.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/Caelum

Ogre3D. 2011b. Current Ogre Features [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/Current+Ogre+Features

Ogre3D. 2011c. Getting Started [viitattu 14.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/Getting+Started

Ogre3D. 2011d. -Mesh [viitattu 14.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/tiki-index.php?page=-Mesh

Ogre3D. 2011e. Ogre Meshy [viitattu 16.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/Ogre+Meshy

Ogre3D. 2011f. OgreXmlConverter [viitattu 29.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/OgreXmlConverter

Ogre3D. 2011g. SceneManagersFAQ [viitattu 29.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/SceneManagersFAQ

Ogre3D. 2011h. Setting Up An Application [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/Setting+Up+An+Application

Ogre3D. 2012a. Licensing [viitattu 14.3.2012]. Saatavissa: http://www.ogre3d.org/licensing

Ogre3D. 2012b. ManualObject [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/ManualObject

Ogre3D. 2012c. OGRE Exporters [viitattu 15.3.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/OGRE+Exporters

Ogre3D. 2012d. Projects using OGRE [viitattu 1.4.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/Projects+Using+OGRE

Orland, K. 2011. Torchlight Sells 1M Copies Across All Platforms [viitattu 1.4.2012]. Saatavissa: http://www.gamasutra.com/view/news/35695/Torchlight_Sells_1M_Copies_Across_All_Platforms.php

Runic Games. 2009. TorchED is here! [viitattu 1.4.2012]. Saatavissa: http://www.torchlightgame.com/gamenews/2009/11/16/torched-is-here/

Runic Games. 2011. An Update from Travis Baldree [viitattu 1.4.2012]. Saatavissa: http://www.torchlight2game.com/news/2011/11/17/an-update-from-travis-baldree

The Linux Information Project. 2009. MIT License Definition [viitattu 22.3.2012]. Saatavissa: http://www.linfo.org/mitlicense.html

Unity Technologies. 2005. Public Relations [viitattu 22.3.2012]. Saatavissa: http://unity3d.com/company/public-relations/press-releases

Unity Technologies. 2012. License Comparisons [viitattu 14.3.2012]. Saatavissa: http://unity3d.com/unity/licenses

Wikimedia Foundation. 2012a. CryEngine [viitattu 14.3.2012]. Saatavissa: http://en.wikipedia.org/wiki/CryEngine

Wikimedia Foundation. 2012b. Unreal Engine [viitattu 14.3.2012]. Saatavissa: http://en.wikipedia.org/wiki/Unreal_Engine

Taulukot

TAULUKKO 1. Rusin, Z. 2009. Open Source Licenses [viitattu 14.3.2012]. Saatavissa: http://web.archive.org/web/20090317083515/http://developer.kde.org/documentation/licensing/licenses_summary.html

TAULUKKO 2. DevMaster. 2011a. CryEngine 3 [viitattu 14.3.2012]. Saatavissa: http://devmaster.net/devdb/engines/cryengine-3

TAULUKKO 2. DevMaster. 2011b. OGRE [viitattu 14.3.2012]. Saatavissa: http://devmaster.net/devdb/engines/ogre

TAULUKKO 2. DevMaster. 2011c. Unity [viitattu 14.3.2012]. Saatavissa: http://devmaster.net/devdb/engines/unity

TAULUKKO 2. DevMaster. 2011d. Unreal Engine 3 [viitattu 14.3.2012]. Saatavissa: http://devmaster.net/devdb/engines/unreal-engine-3

Kuvat

KUVA 1. Wikimedia Foundation. 2006. Unreal Engine Comparison [viitattu 14.3.2012]. Saatavissa: http://upload.wikimedia.org/wikipedia/en/8/81/Unreal_Engine_Comparison.jpg

KUVAT 2-4. Santala, H. 2012. Lahden ammattikorkeakoulu.

KUVA 5. NeoAxis Group. 2011. NeoAxis Engine 1.0 Beta Released [viitattu 1.4.2012]. Saatavissa: http://www.neoaxis.com/screenshots/UpdatedTerrainEditor.jpg

KUVA 6. Runic Games. 2012. Media [viitattu 1.4.2012]. Saatavissa: http://www.torchlightgame.com/wp-content/uploads/1-year-stomp.png

KUVAT 7-13. Santala, H. 2012. Lahden ammattikorkeakoulu.

Kuviot

KUVIOT 1-2. Ogre3D. 2012. ManualObject [viitattu 17.4.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/ManualObject

KUVIO 3-4. Ogre3D. 2011. MadMarx Tutorial 7 [viitattu 17.4.2012]. Saatavissa: http://www.ogre3d.org/tikiwiki/MadMarx+Tutorial+7&structure=Tutorials

LIITTEET

CD, joka sisältää:

- Työssä käytetyt kuvat
- Casessa käytetyt tiedostot